August 9th 2011, OSG Site Admin Workshop

Jason Zurawski – Internet2 Research Liaison

Network Performance Hands-On

Agenda

- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the “first responder”
- Conclusion

Getting into the Machines

- Everyone was mailed a username/password. If not, raise your hand ***now***
- SSH
  - Terminal application for Linux/Mac users
  - Putty (http://www.chiark.greenend.org.uk/~sgtatham/putty/) for windows people.
- How to get in:
  - ssh user@npw.internet2.edu
  - Enter password
- Navigation:
  - 4 hosts (assume direct connections are all-to-all):
    - head
    - red-pc1
    - blue-pc1
    - green-pc1
  - Your SSH key works on all of them
  - Machines are fully configured, ***but the network is broken***
    - We will be finding out how and where it is broken

Agenda

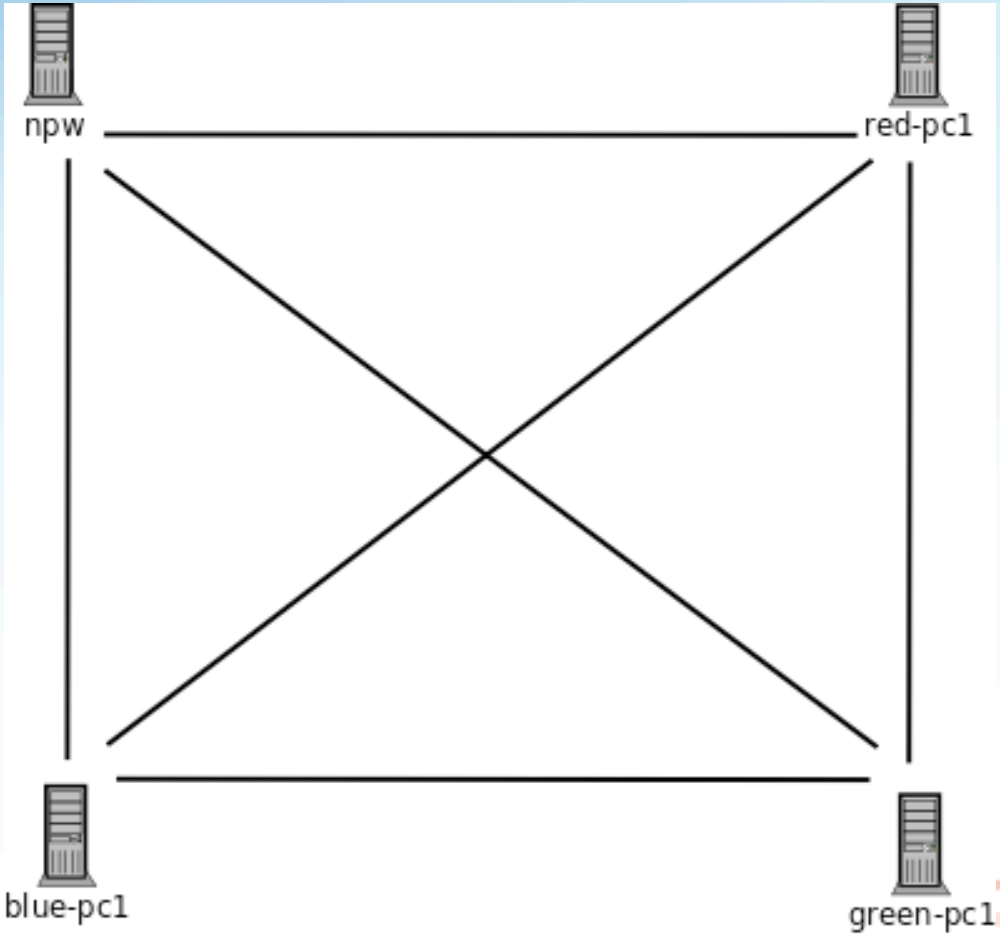
- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the “first responder”
- Conclusion

Exploring Ping for RTT

- RTT = Round trip time.
- This is the time it takes to send an ICMP packet from the source, to the destination, and receive a response
  - ICMP = packet designed for measurement. Note some sites block this …
  - Same sort of data you get from traceroute probes too
- What is RTT good for?
  - "Latency" between hosts. Recall that:
    - Network Latency = Distance Delay + Serialization Delay + Queue Delay + Forwarding Delay + Protocol Delay
  - Rule of thumb:
    - 1ms = ~50 miles
    - Differences in medium (e.g. copper vs fiber) do not amount to much – 5 micro-seconds vs 6 micro-seconds

Exploring Ping for RTT

- 4 Machines, we want to build a ‘RTT’ map of the network. Consider this template:



Exploring Ping for RTT

- How do we do this?
  - Use the 'ping' tool
  - Example use (from head):
    - `ping -c 5 red-pc1`
  - Example results:

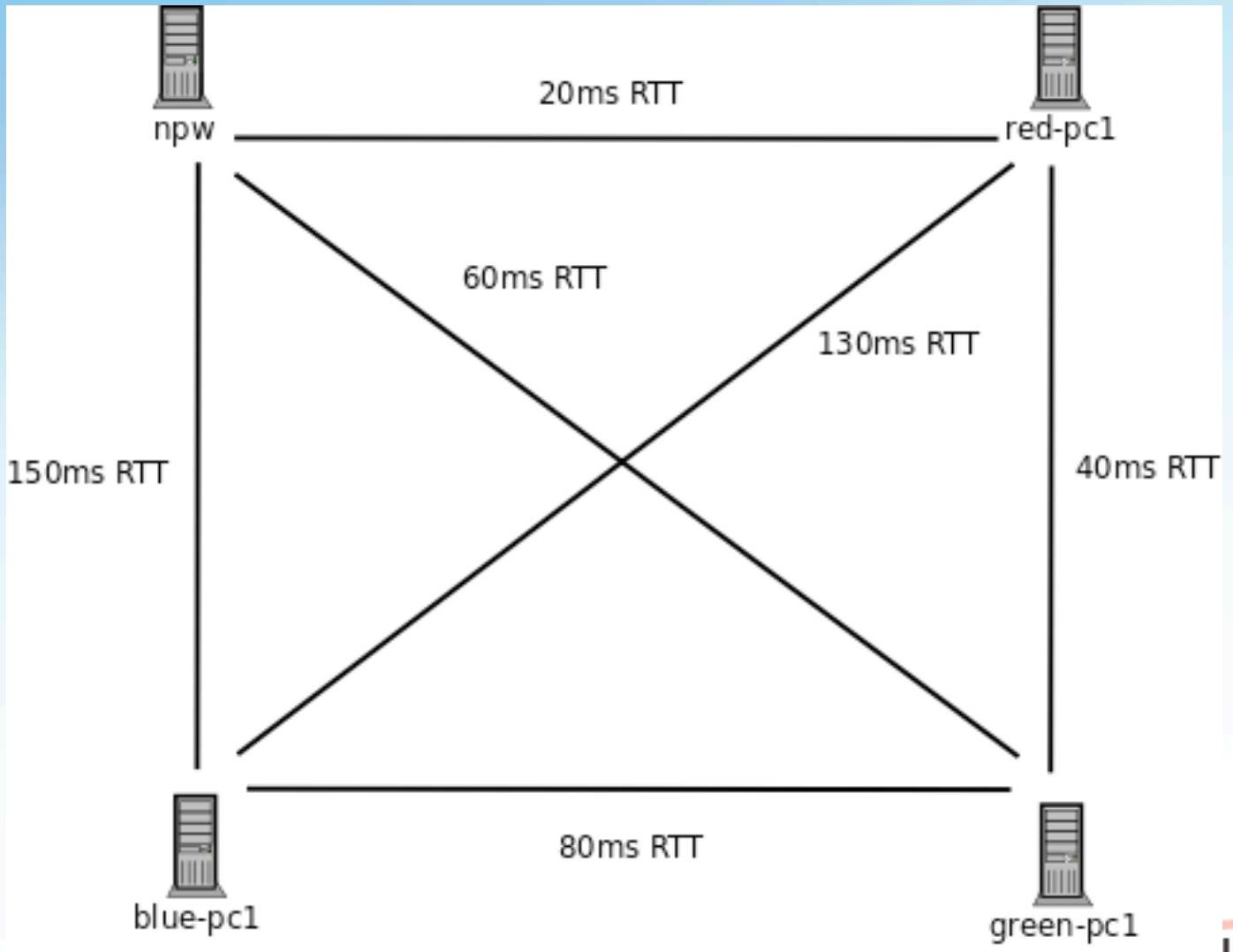
```
PING red-pc1 (192.168.0.2) 56(84) bytes of data.
64 bytes from red-pc1 (192.168.0.2): icmp_seq=2 ttl=64 time=21.0 ms
64 bytes from red-pc1 (192.168.0.2): icmp_seq=3 ttl=64 time=21.3 ms
64 bytes from red-pc1 (192.168.0.2): icmp_seq=4 ttl=64 time=20.6 ms
64 bytes from red-pc1 (192.168.0.2): icmp_seq=5 ttl=64 time=20.6 ms

--- red-pc1 ping statistics ---
5 packets transmitted, 4 received, 20% packet loss, time 4000ms
rtt min/avg/max/mdev = 20.643/20.922/21.341/0.291 ms
```

Exploring Ping for RTT

- What to do:
  - From each host, ping all of the other hosts.
    - From head, ping red-pc1, blue-pc1, and green-pc1
    - From red-pc1, ping blue-pc1, and green-pc1
    - From blue-pc1, ping green-pc1
  - Record the 'average' RTT. Also note any loss or duplication (if you see any).
  - 5 – 10 pings for the '-c' flag is enough for this exercise. Feel free to run more if you want.

Exploring Ping for RTT - Answers



Agenda

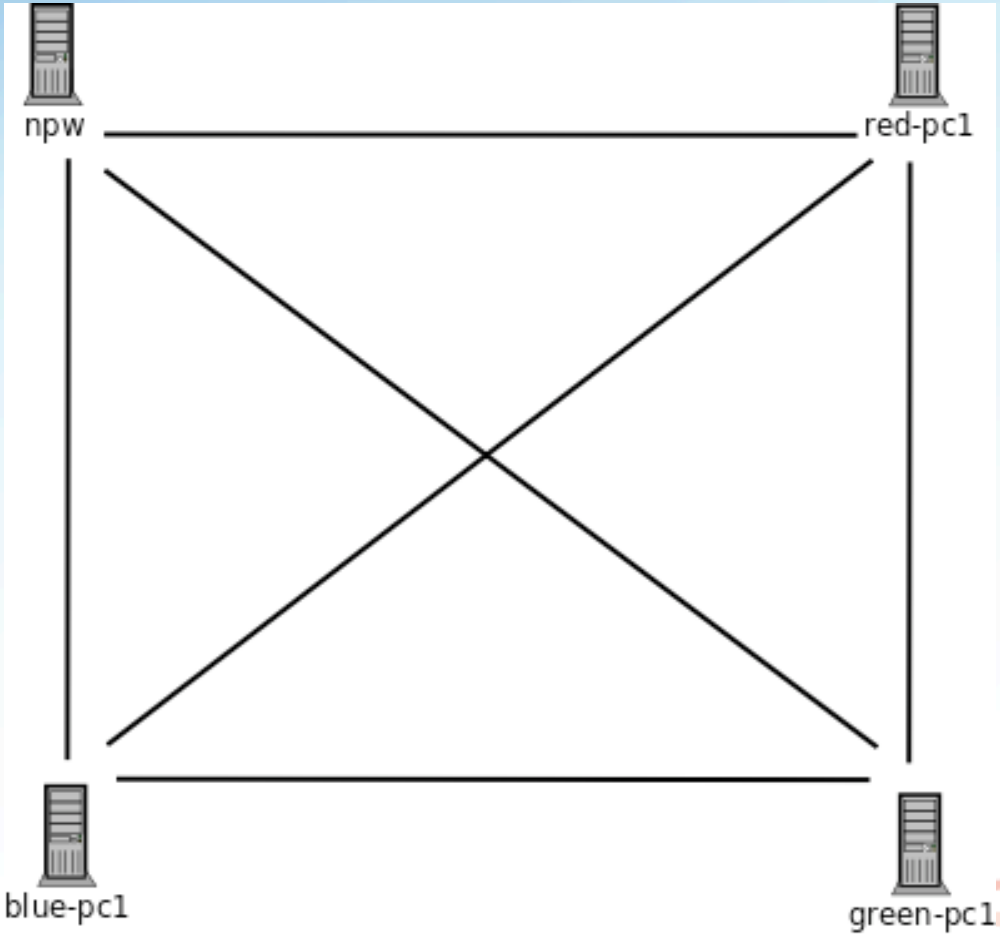
- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the “first responder”
- Conclusion

Exploring OWAMP for OWD/Loss/OOD

- We have RTT, why do we care about OWD (one way delay)?
  - RTT will mask certain network characteristics
    - Hides the 'direction' of a problem
    - Will not reveal delays from queue depths
  - RTT uses ICMP packets, which could be treated differently by routing engines (e.g. ICMP is a lower priority than TCP/UDP, and may be delayed)
- One way delay is not without difficulties:
  - Need a daemon + special client, not simple like ping
  - To do it right, need a standard … and there happens to be one: http://tools.ietf.org/html/rfc4656
  - Need some pretty strict time keeping requirements w/ NTP
- What do we really care about in terms of metrics:
  - One Way Delay
  - Packet Loss
  - Out of Order Deliver of Packets
  - Packet Duplication
  - Packet Jitter (arrival variation)

Exploring Ping for RTT

- 4 Machines, we want to build a 'OWD/Loss/Dupplication' map of the network. Consider this template:



Exploring OWAMP for OWD/Loss/OOD

- How do we do this?
  - Use the 'owping' tool
  - Example use (from head):
    - `owping red-pc1 –c 100`
  - Example results:
    - `Approximately 13.2 seconds until results available`

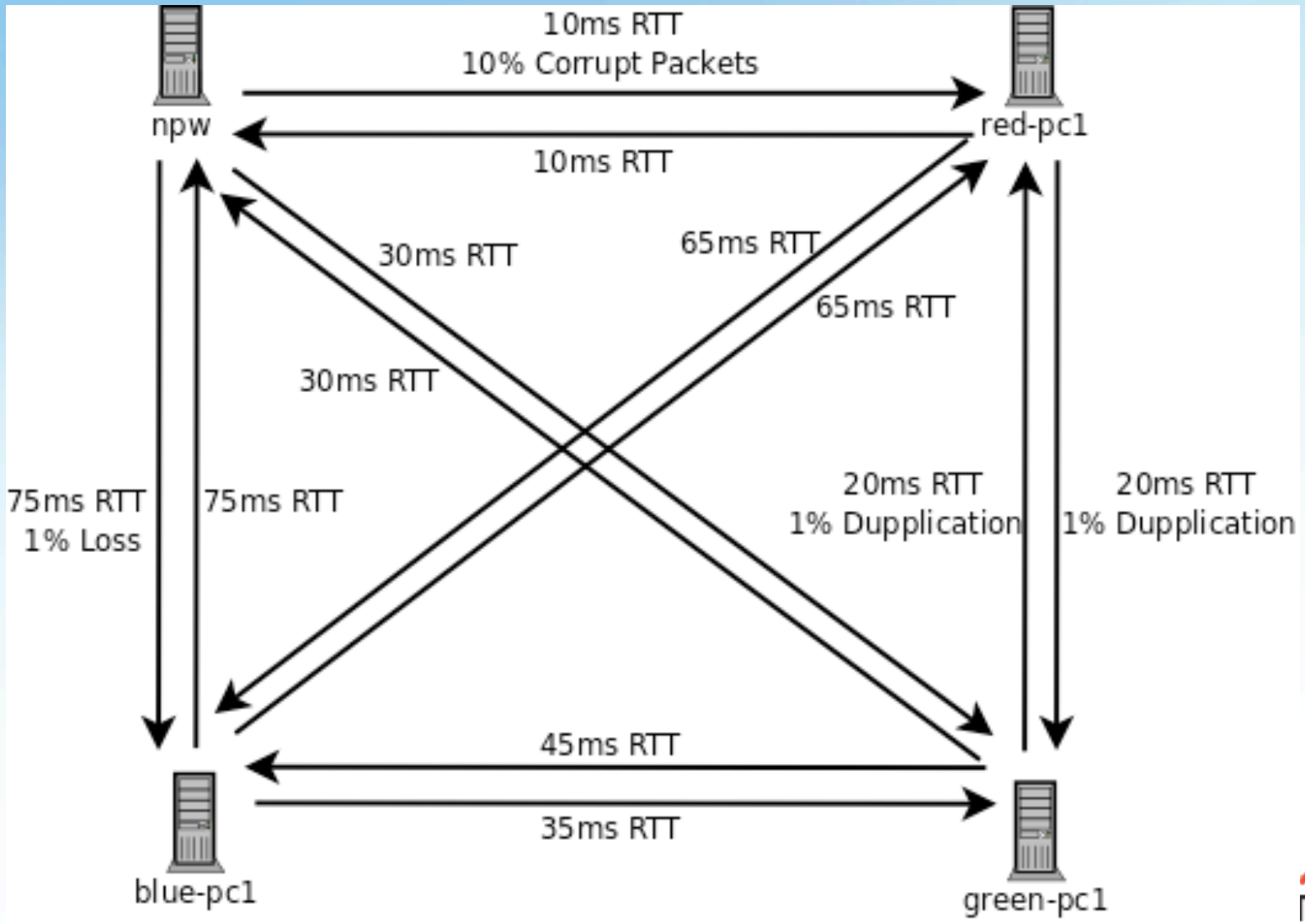
```
--- owping statistics from [head]:44841 to [red-pc1]:43706 ---
SID:    c0a80002d1e551cea21c68ec8801f7e1
first:  2011-08-04T13:13:51.877
last:   2011-08-04T13:14:01.111
100 sent, 11 lost (11.000%), 0 duplicates
one-way delay min/median/max = 10.1/10.3/11.1 ms, (err=0.0751 ms)
one-way jitter = nan ms (P95-P50)
TTL not reported
no reordering


--- owping statistics from [red-pc1]:50122 to [head]:58922 ---
SID:    c0a80001d1e551cea50b417c8529d244
first:  2011-08-04T13:13:51.844
last:   2011-08-04T13:14:03.359
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 10.1/10.4/11.2 ms, (err=0.0751 ms)
one-way jitter = 0.7 ms (P95-P50)
TTL not reported
no reordering
```

Exploring OWAMP for OWD/Loss/OOD

- What to do:
  - From each host, owping all of the other hosts.
    - From head, owping red-pc1, blue-pc1, and green-pc1
    - From red-pc1, owping blue-pc1, and green-pc1
    - From blue-pc1, owping green-pc1
  - Note – there will be two lines of output, one for each direction.
    - We want both, clearly label each
  - Record the 'median' OWD. Also note any loss, duplication, re-ordering (if you see any). Don't record Jitter.
  - 100 pings for the '-c' flag is enough for this exercise. Feel free to run more if you want.
    - 10 packets are sent per second, for a total of 10 seconds (e.g. 100)
    - You may want to run 1000 packets to see events that may be harder to track (e.g. small amounts of loss)

Exploring OWAMP - Answers



Agenda

- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the “first responder”
- Conclusion

Comparison of Latency Findings

- What's the big difference?
  - RTT = OWD_1 + OWD_2
  - But is OWD_1 = OWD_2 ?
  - Sometimes yes, sometimes no. Why?
    - Asymmetric Routing
    - Delays (e.g. propagation, queuing, processing, etc.). Queuing is often an indication of congestion, something that can be an issue in clusters
    - Loss (e.g. failing equipment will force loss of retransmissions). Note that OWAMP can tell us loss numbers though.
- What about the other metrics?
  - Loss?
  - Out of Order Delivery?
  - Duplication?
  - Jitter?

Comparison of Latency Findings

- Loss?
  - All loss is bad. Some loss is worse than others:
    - Loss of data packets = retransmission
    - Loss of ack packets = relying on selective or cumulative acks
  - Path length matters
    - Loss on short RTT paths = easy to recover from
    - Loss on long RTT paths, longer to recover and thus degradation of throughput
  - Loss percentage, 1% vs 10%, which is 'better'?
    - Think about the path lengths in your answer…
- Out of Order Delivery?
  - Packets are delayed/routed differently
  - Stalls a TCP flow because it must 'wait' for missing packets
  - May trigger re-transmissions if the wait is too long
  - Network cards with TCP offload (something designed to help) can introduce this behavior locally

Comparison of Latency Findings

- Duplication?
  - A packet is received on one end, but the 'ack' is lost. The other end re-sends, but this is not needed. Thus we have a duplicate.
  - A device may route the same data packet onto multiple interfaces (e.g. 'punting'), multiple copies arrive.
- Jitter?
  - Variation in the arrival time of packets. E.g. could signify 'bursty' behavior
  - Ideally a packet stream should be well spaced, and the routers can process the data through queues at a constant rate
- The next couple of slides go into some of the nuances of TCP, these will help us better understand what is happening before we move on to the concept of 'throughput'

TCP – Quick Overview

- Data Packet
  - Contains some header overhead, and the broken up chunk of user data
- ACK Packet
  - Acknowledge the receipt of a data packet, “cumulative” in nature
- SACK Packet
  - Selective acknowledgement for a specific missing segment
- MSS
  - Maximum segment size (largest size of packets on a given network segment)
- Congestion Control
  - Process of self regulating flow speed due to loss in the network (e.g. making it fair)
- Slow Start
  - Avoid sending more data than the network is capable of consuming. Goal is to reach a loss (establishes window size by relying on acks)

TCP – Quick Overview

- Congestion Avoidance
  - Additive-increase/Multiplicative-decrease [AIMD] scheme to find a fair speed for a TCP flow by adjusting the sending window. Starts low (2 x MSS) and increase
- Fast Retransmit
  - Retransmit a single segment after receiving duplicate ACKs for the prior numbered segment
- Fast Recovery
  - Re-send packets in a window after a timeout
- Bandwidth Delay Product
  - The amount of "in flight" data allowed for a TCP connection
  - BDP = bandwidth * round trip time
  - Example: 1Gb/s cross country, ~100ms
    - 1,000,000,000 b/s * .1 s = 100,000,000 bits
    - 100,000,000 / 8 = 12,500,000 bytes
    - 12,500,000 bytes / (1024*1024) ~ 12MB

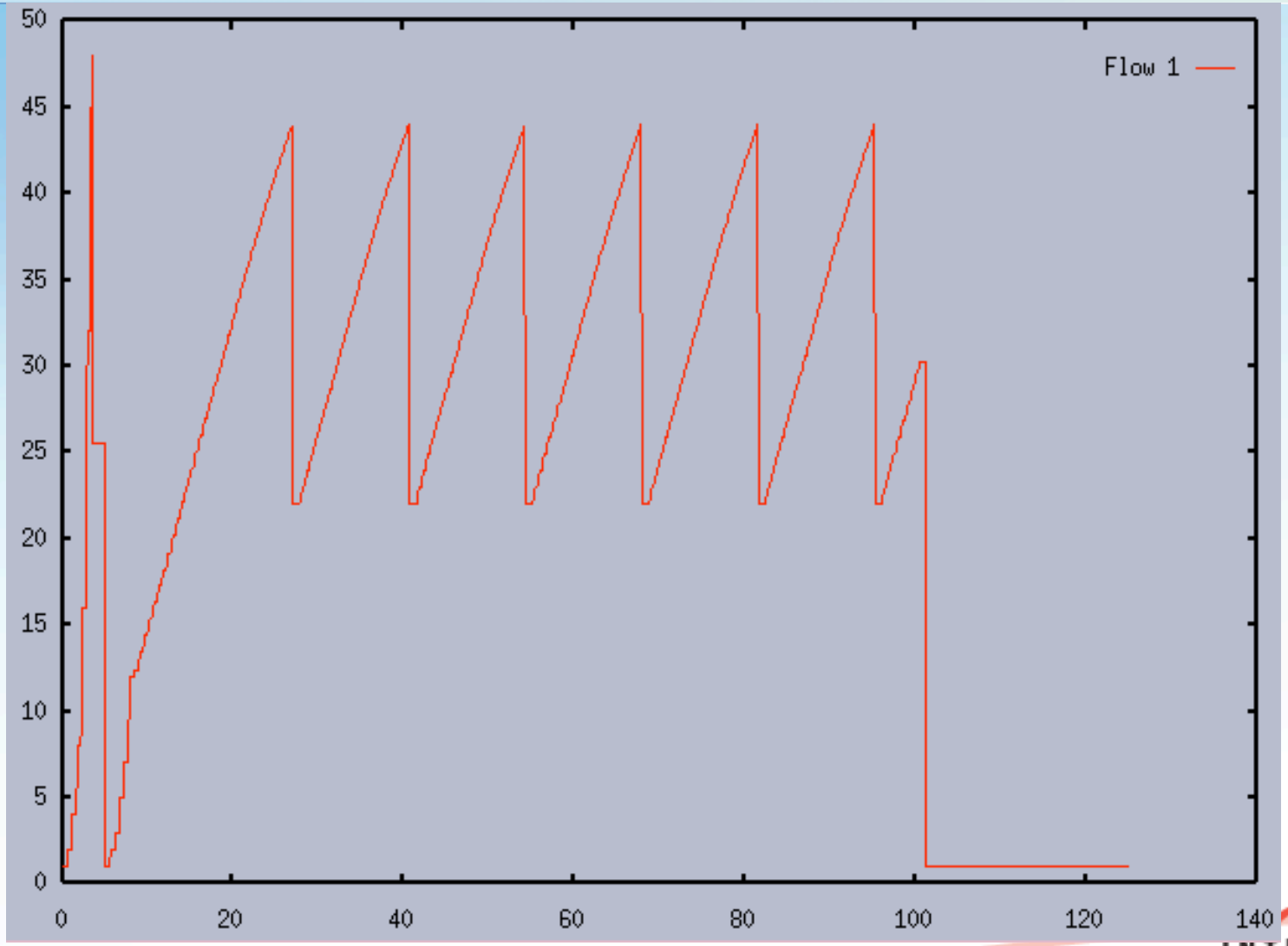
TCP – Quick Overview

- Congestion Control Algorithms (selectable in the Linux Kernel)
  - RENO (Slow Start, Cong. Avoidance, Fast Retransmit, Fast Recovery)
  - Cubic (Optimized for LFN [Long Fat Networks] with large latency, Cubic growth pattern)
  - HTCP (still additive-increase/multiplicative-decrease [AIMD], more agressive as loss decreases on high BDP paths)

TCP – Quick Overview

- General Operational Pattern
  - Sender buffers up data to send into segments (respect the MSS) and numbers each
  - The 'window' is established and packets are sent in order from the window
  - The flow of data and ACK packets will dictate the overall speed of TCP for the length of the transfer

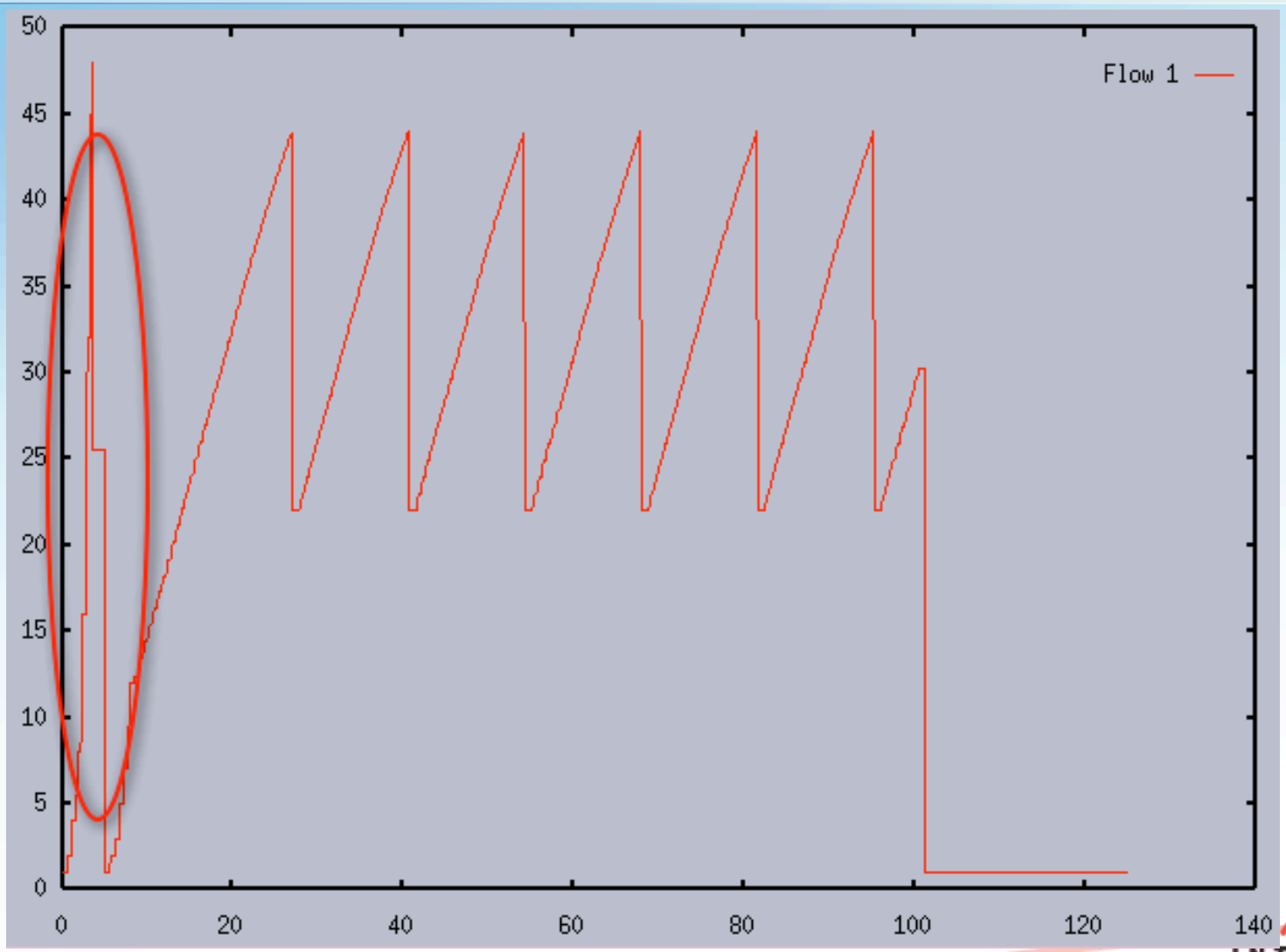
TCP – Quick Overview (Typical Sawtooth)



TCP – Quick Overview

- General Operational Pattern – cont
  - TCP starts slow, until it can establish the available resources on the network.
  - The idea is to grow the window until a loss is observed
  - This is the signal to the algorithm that it must limit the window for the time being, it can slowly build it back up

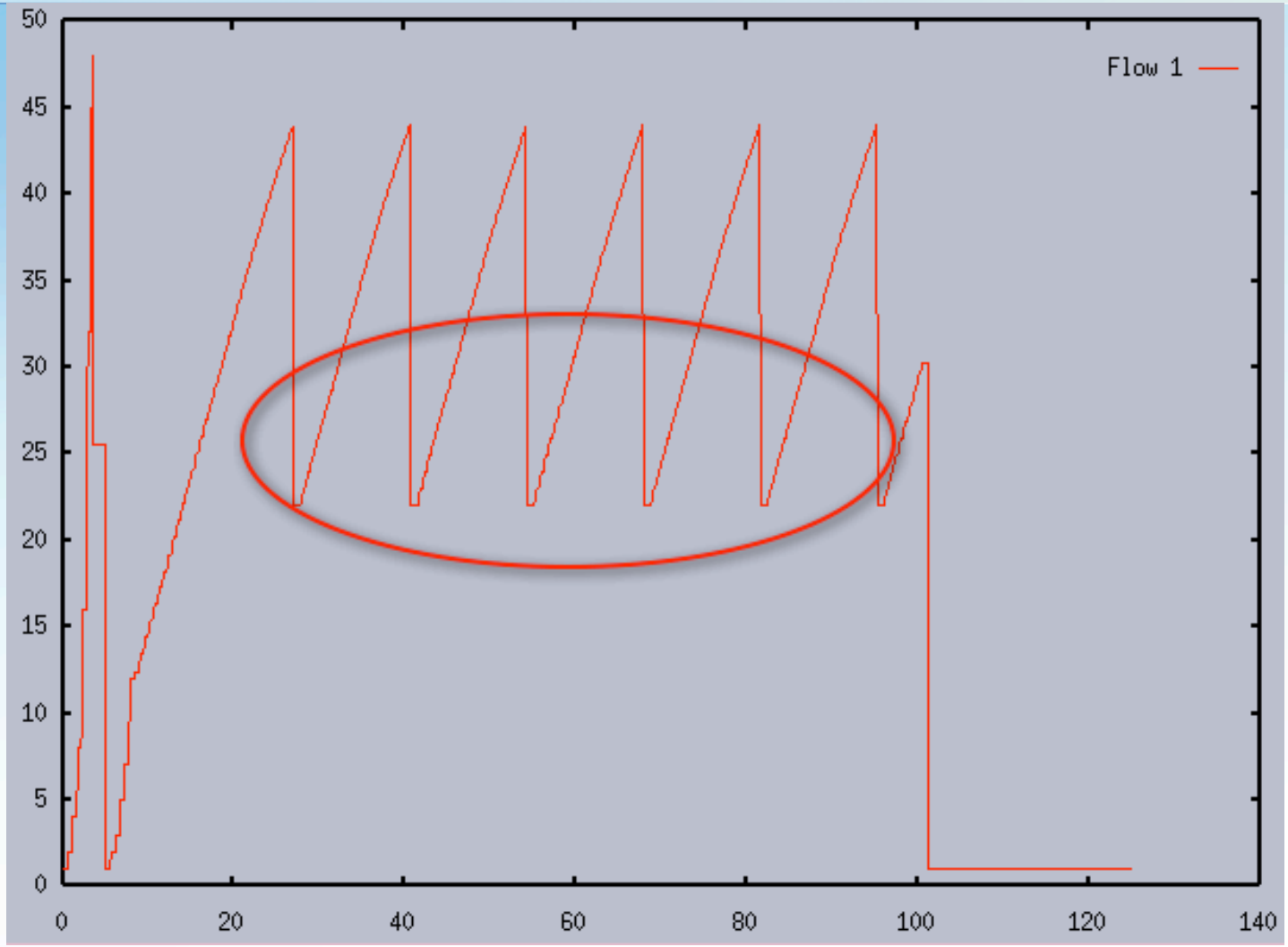
TCP – Quick Overview (Slow Start)



TCP – Quick Overview

- General Operational Pattern – cont
  - Receiver will acknowledge packets as they arrive
    - ACK Each (old style)
    - Cumulative ACK ("I have seen everything up to this segment"
    - Selective ACK (sent to combat a complete retransmit of the window)
  - TCP relies on loss to a certain extent – it will adjust it's behavior after each loss
    - Congestive (e.g. reaching network limitation, or due to traffic)
    - Non-congestive (due to actual problems in the network)
  - Congestion avoidance stage follows slow start, window will remain a certain size and data rates will increase/decrease based on loss in the network
  - Congestion Control algorithms modify the behavior over time
    - Control how large the window may grow
    - Control how fast to recover from any loss

TCP – Quick Overview (Cong. Avoidance)

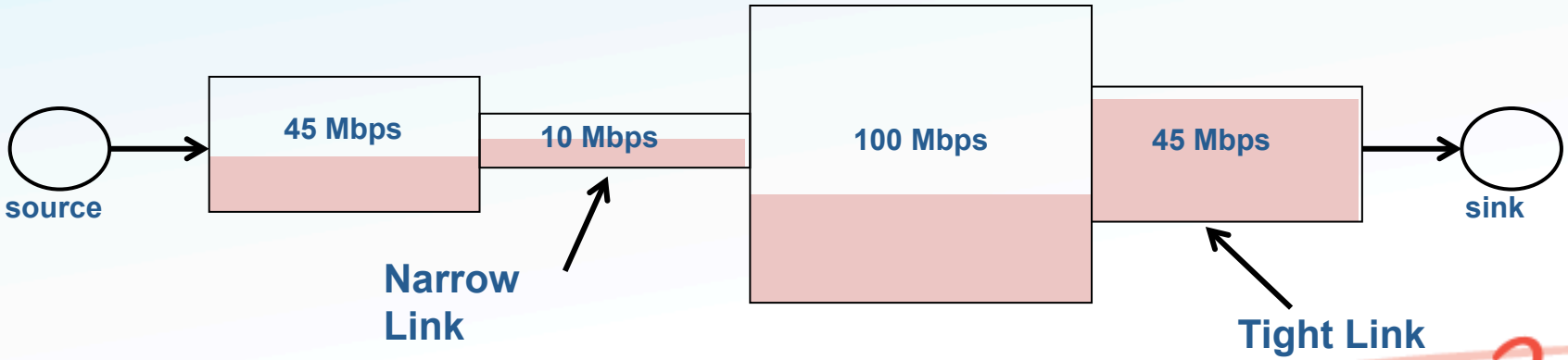


Agenda

- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the “first responder”
- Conclusion

Musings on Throughput

- The term "throughput" is vague
  - Capacity: link speed
    - Narrow Link: link with the lowest capacity along a path
    - Capacity of the end-to-end path = capacity of the narrow link
  - Utilized bandwidth: current traffic load
  - Available bandwidth: capacity – utilized bandwidth
    - Tight Link: link with the least available bandwidth in a path
  - Achievable bandwidth: includes protocol and host issues



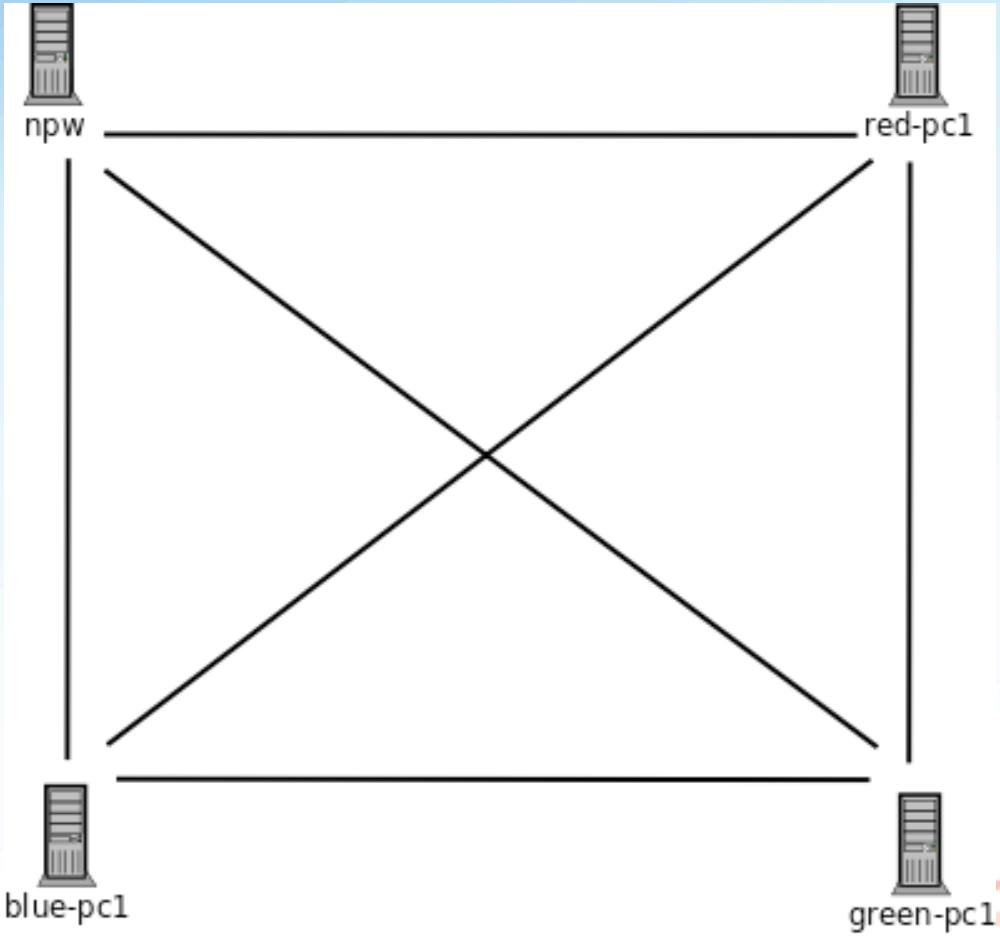
(Shaded portion shows background traffic)

Agenda

- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the “first responder”
- Conclusion

Exploring Throughput

- 4 Machines, we want to build a ‘Throughput’ map of the network like we did before. Consider this template:



Exploring Throughput

- How do we do this?
  - Use the 'bwctl' tool
  - Example use (from head):
    - `bwctl -f m -t 10 -i 1 –c green-pc1`
    - `bwctl -f m -t 10 -i 1 –s green-pc1`
  - Example results:

```
bwctl: Using tool: iperf
bwctl: 17 seconds until test results available

RECEIVER START
bwctl: exec_line: iperf -B 192.168.0.3 -s -f m -m -p 5004 -t 10 -i 1
bwctl: start_tool: 3521471897.344501
------------------------------------------------------------
Server listening on TCP port 5004
Binding to local address 192.168.0.3
TCP window size: 0.08 MByte (default)
------------------------------------------------------------
[ 14] local 192.168.0.3 port 5004 connected with 192.168.0.1 port 5004
[ ID] Interval       Transfer     Bandwidth
[ 14]  0.0- 1.0 sec  3.40 MBytes  28.5 Mbits/sec
[ 14]  1.0- 2.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14]  2.0- 3.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14]  3.0- 4.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14]  4.0- 5.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14]  5.0- 6.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14]  6.0- 7.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14]  7.0- 8.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14]  8.0- 9.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14]  9.0-10.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14] 10.0-11.0 sec  11.2 MBytes  94.1 Mbits/sec
[ 14] 11.0-12.0 sec  11.2 MBytes  94.1 Mbits/sec
bwctl: local tool did not complete in allocated time frame and was killed
bwctl: stop_exec: 3521471912.558740

RECEIVER END
```

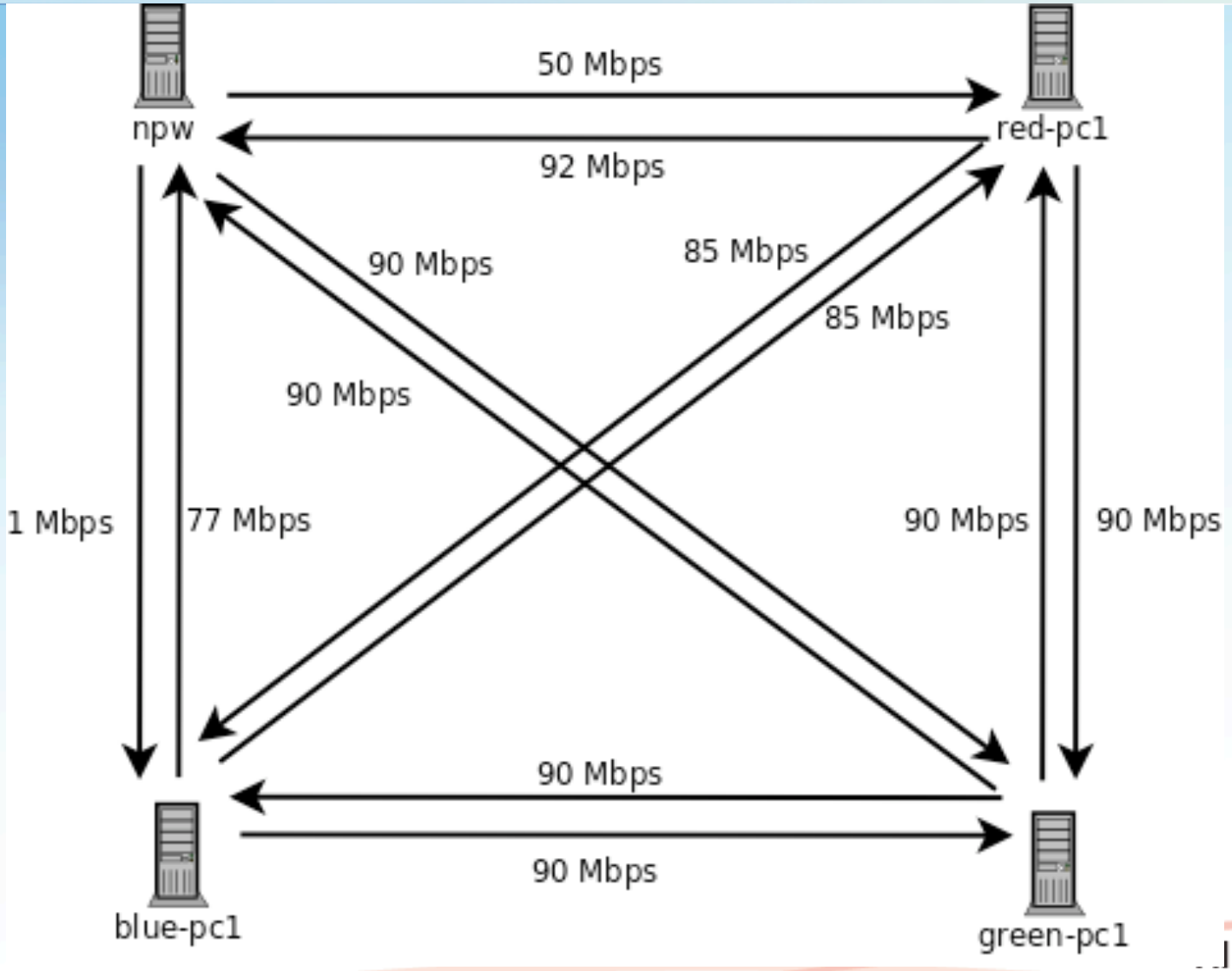
Exploring Throughput

- N.B. BWCTL is a "serial" tool, therefore everyone will be queued when they attempt to test…
- N.B. This is a 100Mb network only – maximum throughput you can expect to achieve is around ~95Mbps
  - ***Why?***
- What to (try to) do:
  - From each host, perform tests all of the other hosts.
    - From head, test to red-pc1, blue-pc1, and green-pc1
    - From red-pc1, test to blue-pc1, and green-pc1
    - From blue-pc1, test to green-pc1
  - N.B. Use the '-c' and '-s' flags in front of the host name, this will alter the direction of the test
    - -c Host = Host is 'catching' (receiving) the data
    - -s Host = Host is 'sending' the data
  - Record the overall throughput.
  - The intervals are useful for determining TCP ramp up/loss

Exploring Throughput

- N.B. We have some regular monitoring in place to get the answers:
  - http://npw.internet2.edu
  - Click on 'Throughput' on the left side.

Exploring Throughput - Answers



Agenda

- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the “first responder”
- Conclusion

Comparison of Results

- Lets focus on certain paths:
  - head -> red-pc1
    - 10% Loss (one direction only, H -> R)
    - Short RTT (20ms), with symmetric (10ms) OWDs
    - Throughput was ~50 and ~90. Why?
      - 50Mbps direction had excessive los, but a short RTT. Can quickly recovery. But always recovering
      - See also the graphs from perfSONAR … erratic behavior
      - 90Mbps direction was not getting ACK packets (getting all data)
  - head -> blue-pc1
    - 1% Loss (one direction only, H -> B)
    - Long RTT (150ms), with symmetric (75ms) OWDs
    - Throughput is ~1 and ~77. Why?
      - 1Mbps direction had little loss, but long RTT. Hard to recover (think of the BDP). Recovery takes a long time, and we cut our throughput in half with each loss
      - 77Mbps direction was not getting ACK packets (getting all data) which still resulted in minor slow down. Need to run a test longer than 10s to see full ramp up (60?)

Comparison of Results

- Lets focus on certain paths:
  - red-pc1 -> green-pc1
    - 1% Duplication, both directions
    - Medium RTT (40ms), with symmetric (20ms) OWDs
    - Throughput ~90 in each direction. Why?
      - Duplication is not as serious as loss. Does not stall a connection as severely. Will end up being idle waiting for ack/data packets on either end occasionally.
      - Small amount is also not noticed on shorter RTT path
  - blue-pc1 -> green-pc1
    - No Loss, Duplication
    - Medium RTT (80ms), with asymmetric (45ms/35ms) OWDs
    - Throughput ~90 in each direction. Why?
      - In this case the asymmetry is not bad (because we are fabricating it). If this was the result of a longer route, or queueing, we would expect to see differences in the numbers.

Agenda

- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the "first responder"
- Conclusion

Using NDT as the “first responder”

- NDT is a single shot tool.
  - Tries to reveal as much about your host and network as it can
  - Will note things like buffer sizes (something you can easily fix: http://fasterdata.es.net/fasterdata/host-tuning/)
  - Will note congestion/bottlenecks between your client and the server
- This is really the first thing you should try when you notice a problem.
  - Use the client on your storage node/compute node.
  - Test against a well known server. Internet2 has 9 of these, pick your closest:
    - ndt.(ATLA, CHIC, HOUS, KANS, LOSA, NEWY, SALT, SEAT, WASH).net.internet2.edu

Using NDT as the “first responder”

- How do we do this?
  - Use the ‘web100clt’ tool
  - Example use (from head):
    - `web100clt -n red-pc1`
  - Example results:
    - `Testing network path for configuration and performance problems -- Using IPv4 address`
    - `Checking for Middleboxes . . . . . Done`
    - `checking for firewalls . . . . . Done`
    - `running 10s outbound test (client to server) . . . . . 58.35 Mb/s`
    - `running 10s inbound test (server to client) . . . . . 93.27 Mb/s`
    - `Server unable to determine bottleneck link type.`
    - `Information: Other network traffic is congesting the link`
    - `Information [C2S]: Packet queuing detected: 50.18% (local buffers)`
    - `Information [S2C]: Packet queuing detected: 14.05% (local buffers)`
    - `Server 'red-pc1' is not behind a firewall. [Connection to the ephemeral port was successful]`
    - `Client is not behind a firewall. [Connection to the ephemeral port was successful]`
    - `Packet size is preserved End-to-End`
    - `Server IP addresses are preserved End-to-End`
    - `Client IP addresses are preserved End-to-End`

Using NDT as the “first responder”

- What to do:
  - Test this out between one host and another host, spread out because NDT will queue tests…
  - Use the ‘-l’, ‘-ll’, and ‘-lll’ options for more interesting output. One ‘l’ will tell you things about your connection and host (which is useful).

Agenda

- Getting into the Machines
- 1st Exercise – Exploring Ping for RTT
- 2nd Exercise – Exploring OWAMP for OWD, Loss, OOD
- Comparison of Latency Findings
- Musings on Throughput
- 3rd Exercise – Exploring Throughput
- Comparison of Results
- 4th Demonstration – Using NDT as the “first responder”
- Conclusion

Conclusion

- Thanks for listening to the brief introduction to performance measurement
  - Its hard to condense things into an hour.
  - If you think this sort of thing would be good for your lab/campus, drop me a line – zurawski@internet2.edu
- All of these tools are available in the OSG VDT package, and are useful when things just don't seem right with performance
- Don't suffer with poor performance, there are lots of people available to help fix things

Network Performance Hands-On

August 9th 2011, OSG Site Admin Workshop

Jason Zurawski – Internet2 Research Liaison

For more information, visit http://www.internet2.edu/workshops/npw